A Progressive Query Materialization for Interactive Data Exploration

Vikram Singh1, SK Jain1,

1 Computer Engrg Dept, National Institute of Technology, Kurukshetra, Haryana-136119, India
viks@nitkkr.ac.in, skj@yahoo.com

Abstract. Data analysis is task of competently digging out data insights even if user is uncertain but database systems unable to meet. User’s Iterative interactions with system might be an alternative. Interactive data exploration (IDE) is one such system, supports exploration by incorporating user intention. IDE is key ingredient of many discovery applications, such as scientific computing, financial analysis, Social Analytics etc. IDE supports user's navigation via query to query transition, via ‘exploratory session’. A session often consists of long, complex and analytical queries, when processing against a large and multi-faceted data, hence consume huge time. Although, decomposing an original query into ‘checkpoint queries’ and materialization can significantly reduce time, extensively used in query navigation. We proposed, checkpoint queries selection in greedy manner considering various heuristics. Although, selection of checkpoint queries and improving query reuse are two key challenges in this process.

Keywords: Checkpoint Query; Interactive Data Exploration; Data Analytics; Query Lattice; Query Reuse; Query Result Overlap Ratio

1 Introduction

Typically, users interact with a database system by formulating queries. This query-result mode of interaction assumes, that users are to some extent familiar with the content of the database and that they have clarity on their information needs. However, as databases systems are growing and accessible to a wide spectrum of users, from a domain expert to a non-technical or a naïve user. A typical technical user often assumed to be familiar with the database semantics or structure. The priori knowledge about the database statistics helps user in query formulation of the initial query for interest of the data. Data exploration is the key component in many discovery-oriented application domains, such as scientific analysis, evidence based medicine, financial analysis and social analytics etc. The database structure in such domains is often complex, as the data is captured via multiple medium. An expert user faces a weighty task to formulate exploration query in such complex database

1 Please note that the LNCS Editorial assumes that all authors have used the western naming convention, with given names preceding surnames. This determines the structure of the names in the running heads and the author index.
systems. Thus a system is required, for user assistance in the data exploration by either simply suggesting some queries [1],[2],[3] or some help in query formation.

In data exploration, initially user imposed simple query and subsequently reformulate. This querying paradigm is incremental and iterative, thus a query is constantly evolving through a cycle of intent-query-execution-result. For example, citizen table in a country database having more than a million tuples and a social scientist wants to find interesting facts like living standards and lifestyle of senior citizens in a particular region. A query select * from citizen where age > 60, would return millions of records and highly tedious for infer any conclusion. A fundamental data exploratory system generates few query suggestions [3]. The suggestion can be list of queries like, select * from citizen where age between 60 and 70 and income < 5 lakh or select name, address, contact_details from citizen where has_relative= “no”, or, select * from citizen where location “downtown”. The initial query would be very expensive, based on the suggested queries and result set the user can focus or defocus using tools [21],[22],[23],[27], referred as guided interaction [28].

Data exploration (DE) is highly ad-hoc, interactive process and remains as a resource-and labor-intensive task. Despite its growing importance as current DBMSs cannot effectively support such interactive, multi-step tasks with imprecise goals, as shown in figure 1. In DE, user continually focusing and defocusing over data space to reach up to the relevant part, this navigation is implemented via query-to-query transition, also called query steering. In order to retrieve data objects of real interest, it is highly probable that a user review same query result set multiple times [10]. In an exploratory session, the possibility of result overlap results into query reuse [9].

**How to reuse this result overlap among queries in data exploration** is thus, an interesting research problem, which we explore in this paper. Checkpoint queries are materialized as views, from user queries set with high result set overlapping within exploratory session. Later, in a data exploration session, these views acts as data source to retrieve the relevant data items for user query or use.

In this paper, we devised an approach for reusing the overlap among queries in ‘exploratory session’, by simply materializing the previous queries. The selection of reusable queries ‘checkpoint queries’ is based on the result overlap with the recent or past queries in a ‘Query Session’. For a user exploratory session top-k queries are identified based on heuristics, query result overlap, query frequency etc. Materialized views are often pre-computed and store summarized information thus reduces overall response time by an analytical or exploratory queries. Materialized views (from checkpoint queries) are used during the query reformulation in query steering for data exploration as data source.

![Fig. 1. Interactive Data Exploration (IDE) system](image-url)
1.1 Related Work

The materialization of queries on the historical data is a frequent and familiar approach in the data warehouses [11],[17]. Most of the existing view selection approach considers frequency, size of views, constraints like maintenance cost, time, and disk space as common heuristics to decide which query is selected for materialization. The materialized view selection issue has been investigated in several contexts: query optimization, warehouse design, data placement in a distributed setting, web databases, etc. Many diverse solutions to the view selection problem have been proposed and analyzed through surveys [13],[14],[15],[16]. The survey in [15] concentrates on methods of finding a rewriting of a query using a set of materialized views. The study presented in [14],[16] focuses on the state of art in materialization for web databases. Similarly, an analysis materialized view selection in data warehousing is provided [13]. A complete survey on view selection problem and identification of view selection dimensions along with view selection methods have been classified in [12]. A progressive algorithm for selecting views for materialization is proposed in [20].

However, in the exploratory search based applications either similar queries or expanded query is used repeatedly. An IDE system itself guides the user to browse through the data to find meaningful pattern. [4],[5],[6],[8] exploits the knowledge of data to help non expert users to construct exploratory queries. Query is modified by the user to produce desired result according to his requirements [24]. The concept is similar to that of web search where the search engine guides the user form imprecise queries to relevant web pages. The result further steer the user to drill deeper in the query and modify search terms to get sufficient answers [25]. [7] Defines the query steering algebra like DRILLDOWN and RELATE for navigation in the query sessions. [26] Suggests query manipulations for performance benefits. The goal of data exploration is to provide users as much insight in data as possible. In this intention, a user re-explores similar data space repeatedly by overlapping query. Hence, a paradigm is required to exploit the query overlap. Queries are constructed for exploratory research using sample queries in [18],[25]. For Big Data [19] offers one interesting solution. In some case lattice can be used to represent multi-dimensional data. [29],[30] highlighted how to manipulate a lattice framework for data analysis. In our work, lattice is constructed for selected query set based on heuristics and further finalized for the materialization according to the benefit values.

1.2 Contribution and Outline

An approach for query steering for interactive data exploration system is the prime contribution of this paper. The approach is adapting the notion of view materialization of frequent queries posed by user in query session or exploratory session. Therefore, the repetition of the sub-queries or checkpoint queries is more likely. The reuse of materialized checkpoint queries for data exploration is primary motivation of the approach. In this paper, section 2 discusses the interactive data exploration systems fundamentals and a formal system model is illustrated. Section 3 contains proposed approach for the checkpoint query generation. Identification of frequent queries for
materialization is described in the subsequent subsections. In section 4 we have highlighted various challenges in the current approach. The conclusion is the brief review on current work discussed in the paper.

2 Interactive Data Exploration

The purpose of data exploration is to extract unknown/hidden insight. In Interactive Data Exploration (IDE) the user starts by asking highly selective queries and subsequently refining the queries based on reviewing the obtained result set. In this case the user is not capable of narrowing down his interests by understanding the behavior of the system; hence he is highly dependent on the system to come up with the solution. In this case the system suggest by giving “You may also like” features [2]. On the contrary, when IDE is viewed as a process the user can learn more about his requirements by browsing through the result sets. After a few iterations the user is able to drill down his interests to the point where he is very specific about his data requirements. In this case the process works by giving ‘Did you mean’ features. In this paper we are building IDE as a process.

The data exploration system is similar to a magical crystal ball. The exploratory user can “see into” the final results and uses the information to change the current exploratory process by adjusting the current operation or by issuing the new data request. It is observed that people tend to submit the same queries or slightly modified versions or earlier submitted queries [9]. Such observations involving high and overlap substantial reuse in the information needs of user forms the basis of our query check-pointing framework. In this paper an approach for query reuse is proposed, since user queries with high degree of overlap are most likelihood to appear again in exploration, thus it would be beneficial if these frequent queries are pre-computed and materialized.

2.1 System Model

The workflow of the proposed system is shown in figure 2. The system assumes that the checkpoint queries are valid for a single exploratory session and queries are overlapped. Our proposed approach utilized this overlap among user queries and identifies optimal set of such queries, referred as checkpoint query, concurrently. Checkpoint queries are used in subsequent exploration task either as reformulation or data source, significantly improve exploration in exploration. Since there can be a large number of input queries, it is wise to reduce the number of input queries space based upon some heuristics. The first heuristic is the frequency of query asked.

The first phase starts by applying the criteria’s frequency of queries and size of the result set to select optimal number of queries. The size of the result set is a measure of how expensive it would be to materialize the query and larger queries Fig. 2: Checkpoint Query generation for IDE.
are not preferred since their maintenance cost is high too. Now, these queries are posed for the query lattice construction. In this phase lattice is constructed where queries act as nodes. An iterative process selects top-k queries from query lattice constructed, according to Net-Gain and queries with positive Net-Gain are identified as checkpoint queries. Note that this phase is iterative, until it selects k candidate queries. Multiple checkpoint queries sets are created after each iteration.

3 Query Lattice and Query Result Overlap Ratio (QROR)

In proposed framework, dependencies among related queries are represented in a lattice structure. A lattice node represents individual query and semantic relation between pair of queries is equivalent to edge.

3.1 Lattice Construction

The selected query set after applying two heuristics are now mapped into a lattice structure. In our approach, queries are mapped into lattice structure in a two phase process. Phase 1, simply evaluates the attribute similarity in pair of queries and subsequently phase 2 resolves the appropriate position for a query in the lattice. The key challenge in this is to find the appropriate relation between the queries, such that a given set of queries forms a lattice, another challenge is how to resolve the conflicts in lattice construction. For example, two user queries \(Q_1\) and \(Q_2\) and \(Q_1(A,B) \leq Q_2(A,B,C,D)\) represents \(Q_1\) can be answered using the query \(Q_2\) result set as accessed similar attributes, \(ABCD\) of relations. Exploratory search, we assumed that in an exploratory session query predicate are consistent in query-to-query transition, thus two queries accessing similar attributes may have overlap result set. This notion of query attribute similarity (QAS) is basis of query lattice construction. Some cases two queries are competing for a lattice node; the conflict is resolved by using query result overlap among query pair. For example, two queries \(Q_3(B,C)\) and \(Q_4(B,C)\) accessing similar attributes. The query with higher QROR (Query Result Overlap Ratio) than parent nodes acquires its position in the lattice.

Phase 1: Assume that all the attributes in a relation schema is \((A,B,C,D)\) accessed by the input query. Query nodes at higher level are the ones that access more attributes than the lower level query nodes. The notion of Query Attribute Similarity (QAS) is simply gives the measure similarity among pair of query nodes in lattice of attribute. In as \(Q_3(A,B,C)\) has a higher QAS values with \(Q_4(A,B,C,D)\) than \(Q_4(B,C)\).

Now, Meet is defined as the common of attributes accessed in two queries. So, Meet defined as intersection of attribute pair the \(\{A, B, C\} \land \{A, C, D\} = \{A, C\}\) and Join is combination of all the attributes accessed via pair of queries. Like, \(\{A, B, C\} \cup \{A, C, D\} = \{A, B, C, D\}\): Query lattice can be depicted using the Hasse diagram, as shown in figure 3.

![Fig. 3. Hasse Diagram of user queries in an Exploratory Session](image-url)
Phase 2: If we have two candidate queries competing for their appropriate position in lattice. A query with higher QROR value with their parent (Join Query) is selected. As QROR measures the similarity in the result set between query pair, higher the similarity more the overlap between. For example, two queries Q₂(A,C) and Q₃(A,D) their join is the query Q₁(A,C,D) and two queries Q₂ and Q₃ competing for the nodes neighbor to Q₁. If it is found that QROR of Q₁ to Q₂ is higher than Q₁ to Q₃, then Q₂ is selected as the node in the lattice instead of Q₃.

3.2 Query Result Overlap Ratio (QROR)

The overlap among the queries in an ‘exploratory session’ is an inherent feature in exploratory search. A user in start his exploration journey from a point and navigates by exploring neighborhood regions. This progressive exploration often based on query-to-query navigation. In data exploration systems materialize results overlap can profitable in exploration performance. In proposed approach, we have utilized this query result overlap ratio (QROR), as the decision parameters for the degree of materialization. QROR simply indicates the propositions of cardinalities of query pair (Qᵢ, Qᵢ₊₁). For a given query Q₁ and Q₂, Let φ|Q₁| and φ|Q₂| denotes the search results and # denotes cardinalities of the search result. Query Result Overlap Ratio (QROR) for the two queries is,

$$QROR(Q₁,Q₂) = \frac{\#(φ|Q₁|\capφ|Q₂|)}{\#(φ|Q₁|)}$$ (1).

3.3 Checkpoint Query Identification

Once lattice is constructed, the iterative process generates checkpoint queries to be materialized. The Iterative selection algorithm called Query Checkpoint for Reuse (QCR), which evaluates the advantage, cost and net gain of each query for selection. A positive Net-Gain indicates a perfect candidate query for materialization. Advantage of a user query (Q), is discussed below, where, size of a query is simply the cardinality and parent of Q is the lowest upper bound query from which the result set of Q can be derived.

$$Advantage (Q) = \left( \frac{\text{Size}(\text{Parent of } Q \text{)} - \text{Size}(Q)}{\text{Number of Records per Block}} \right) \times \text{Frequency (Q)} \times \text{Time of one Block Access} \quad (2)$$

$$Cost = \text{Init. Cost} + \text{Search Cost} + \text{Process Cost} \quad (3)$$

$$\text{Search Cost (Q)} = \text{(Selectivity factor)} \times T (\text{Block Retrieving Cost}) \times \text{Size (Q)} \quad (4)$$

The Cost of Execution (Q) as follows and Init Cost are cost of initialization and Search Cost is simply cost to process the selected tuples. Each iteration of the QCR algorithm, queries with maximum Net-Gain is selected. A query with positive Net-gain value becomes the checkpoint queries and thus selected for materialization. Negative Net-Gain of a query is results into next iteration; Net-Gain is defined as,

$$\text{Net Gain (Q)} = \text{Advantage (Q)} - \text{Cost (Q)} \quad (5)$$

The QCR module takes these queries as input and produces a set of candidate checkpoint queries (CQ)= {Q₁, Q₂,...,Qₙ} as output. Each run of QCR module create
check-point queries sets \{CQ_1, CQ_2, \ldots, CQ_m\} based on the queries issued after the set CQ_{t-1} already being constructed and before the set CQ_t, as depicted in Figure 4.

![Diagram](https://via.placeholder.com/150)

Fig. 4: Multiple Checkpoint Query Sets for Original Query Q

### 3.4 Example

Consider the database below and an exploratory session with 100 queries, selecting most frequent queries and based result set of queries Q_1 to Q_2 in the current session. The next phase calls for construction of lattice on the basis of attribute similarity as explained in prior section assuming a single user. Consider a user queries, (Q_1): “Name, Email_id, course_id and building_id of the students taking a course that is taught in the Sector 1, 2 and 3”, (Q_2): “Name and Course_id of students taking a course that is taught in the Sector 1”, (Q_3): “Email id and Building_id of the students taking a course that is taught in the Sector 1 and 2”, (Q_4): “Building_id of all the buildings in the Sector 1” on below given DB.

<table>
<thead>
<tr>
<th>Student</th>
<th>Enroll</th>
<th>Building</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stud_id</td>
<td>Name</td>
<td>email_id</td>
</tr>
<tr>
<td>CO3103</td>
<td>Adam</td>
<td><a href="mailto:adam@abc.com">adam@abc.com</a></td>
</tr>
<tr>
<td>CO3104</td>
<td>Alice</td>
<td><a href="mailto:alice@abc.com">alice@abc.com</a></td>
</tr>
<tr>
<td>CS3113</td>
<td>Bob</td>
<td><a href="mailto:bob@abc.com">bob@abc.com</a></td>
</tr>
</tbody>
</table>

The user queries access set of attributes, such as Q_1(name, email_id, course_id, building_id), Q_2(name, course_id), Q_3(email_id, building_id), and Q_4(building_id). The lattice is defined on above queries, also shown below, Now, user issues a new query (Q_5): “Building_id of all buildings in the Sector 1 and 2”, due to which a conflicts occurs. Now, to resolved this conflict QROR(Q_3, Q_4) and QROR(Q_2, Q_5) is considered. Number of rows in Q_3 is 3, Q_4 will be 1. Since it has a limiting condition in its where clause, and in Q_5 it will be 2. QROR(Q_3, Q_4) is 0.5 and QROR(Q_2, Q_5) is 1. Hence conflict resolution will be in favor of query Q_5 thus Q_5 will replace Q_4 in the lattice structure.

Once lattice is constructed, top-k queries are selected iteratively. Assumptions: initial query setup cost and the cost of processing of each tuple a 1 and block size of 16 bytes. For query Q_3, number of tuples returned will be 1, related evaluations are shown in below table. Since the Net-Gain(Q_3) > 0, qualifies the conditions

<table>
<thead>
<tr>
<th>Advantage(Q_3)</th>
<th>Cost(Q_3)</th>
<th>Net-Gain(Q_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>((</td>
<td>\text{Size(Parent)} = 3</td>
<td>-</td>
</tr>
</tbody>
</table>
required to become a checkpoint query and materialized. Now, when \( Q \) query is materialized as follows: if suppose, a query later issued by a user is in the form of “Find the emails of students who will be moved to B-3 since sector 1 is under renovation”.

### 3.5 Checkpoint Threshold

To maximize the performance benefit of the system, we mandate to have multiple numbers of query checkpoint sets, created after a definite interval termed as checkpoint threshold \( \partial \). The number of queries in every set depends upon the heuristics like number of queries issued by the user, memory constraints, QROR values calculated, SM similarity found etc. If \( k \) is the number of queries in the checkpoint set then \( k \) can be variable for each checkpoint query set. The value of \( k \) can be either fixed or variable depending upon the mode in which QCR is implemented. The checkpoint threshold \( (\partial) \) can be either pre-decided or during the session. We propose dynamic \( \partial \), since views materialization is done on-the-fly. The interval between checkpoint queries is determined dynamically along with the exploration progress. In some case, \( \partial \) values are calculated according to number of queries issued so far and value QROR. The queries obtained, SM similarity generated, constraints like memory etc. Check pointing modes are described below,

<table>
<thead>
<tr>
<th>Mode</th>
<th>Commands</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic</td>
<td>Set_Threshold (&quot;value&quot;)</td>
<td>To Set the value of ( \partial ) before the session begins as in automatic mode</td>
</tr>
<tr>
<td>Variable</td>
<td>Reset_Threshold (&quot;value&quot;)</td>
<td>To alter the value of ( \partial ) during the ongoing user session as in variable mode</td>
</tr>
<tr>
<td>Manual</td>
<td>Checkpoint Ci (Q)</td>
<td>To make a query Q as checkpoint in the manual mode</td>
</tr>
</tbody>
</table>

### 4 Analysis and Challenges

The proposed approach of query reuse for query steering despite of its simplicity gives an effective way of answering the exploratory queries in data exploration systems.

**Optimal Checkpoint Query set.** Finding an optimal set of checkpoint queries helps to answer the question of *how much we used reuse* the previously issued queries. In an exploratory session, user issued large number of related queries. In our approach the database engine makes checkpoint queries till it reached a fixed interval, according to checkpoint threshold \( (\partial) \). After threshold \( (\partial) \) interval of exploratory session another checkpoint set must be created based upon the queries issued after the last stable checkpoint set. The number of such sets should be decided in the basis of the resources available, length and requirements of user session.

**Selection of Checkpoint queries.** Another challenge is the selection of the checkpoint queries to be materialized. It answers one of the most critical question of finding out *which queries should be selected for reuse*. We devised benefit and advantage to derived net-gain for each query to indentify the queries to materialize as views. Primarily, net-gain value gives sufficient basis for selection, as it is based on cardinality of queries. In order to improve the query selection accuracy query semantic or structure or query graph can be used.
Materialized Views Selection for Query reformulation. In the proposed approach, a checkpoint query is materialized as views and later in a user query these views are replaced by a sub-query of the exploratory task. Checkpoint query will minimized query data retrieval and re-computation of the some part of query. View selection will also help to determine the degree of query reuse. Data exploration as an iterative system of query processing and data retrieval, thus to serve the query reformulation using materialized view becomes an online view selection problem. The materialized views are selected on the fly fashion in query reformulation. The goal is to select an appropriate set of views that minimizes query response time and cost of maintaining the selected views, given limited amount of resource e.g., materialization time, storage space.

Appropriate Checkpoint Threshold (ζ) mode. In a exploratory session, checkpoint threshold sets the milestone for query to consider for selection and further materialization. Thus appropriate application of threshold modes in a exploratory session gives a sufficient number of user queries. In manual mode, a user marks the threshold point according to his awareness, while in automatic mode systems automatically imposed the threshold point in session.

5 Conclusion

In interactive data exploration (IDE), data exploration is data-driven and interactive; as a database user iteratively apply his intention of data objects, each user interaction typically serves as a navigation-off point to the next. IDE rarely involves independent, entirely ad-hoc query sequence. Each query session typically consists of overlapped queries. The ‘overlap’ in queries, introduces opportunities for the query reuse. This progress in exploratory search can utilized to reduce the processing effort in future exploratory queries. The result overlap among Qi and Qi+1 is an important indicator for making decision ‘How much to reuse’ and further in candidate query selection ‘which queries to be reuse’. Our proposed work is intended to deal with both issues to an extent. Firstly, it identifies the list of checkpoint queries in a user session for materialization, which addresses the first concern. Another issue is addressed by employing the QOR and query frequency value for the selection of reusable queries. Our strategy promotes and maximizes the reuse of progress in future exploratory session by user. The checkpoint queries sets obtained will be used to explore the data space further by the user.

References